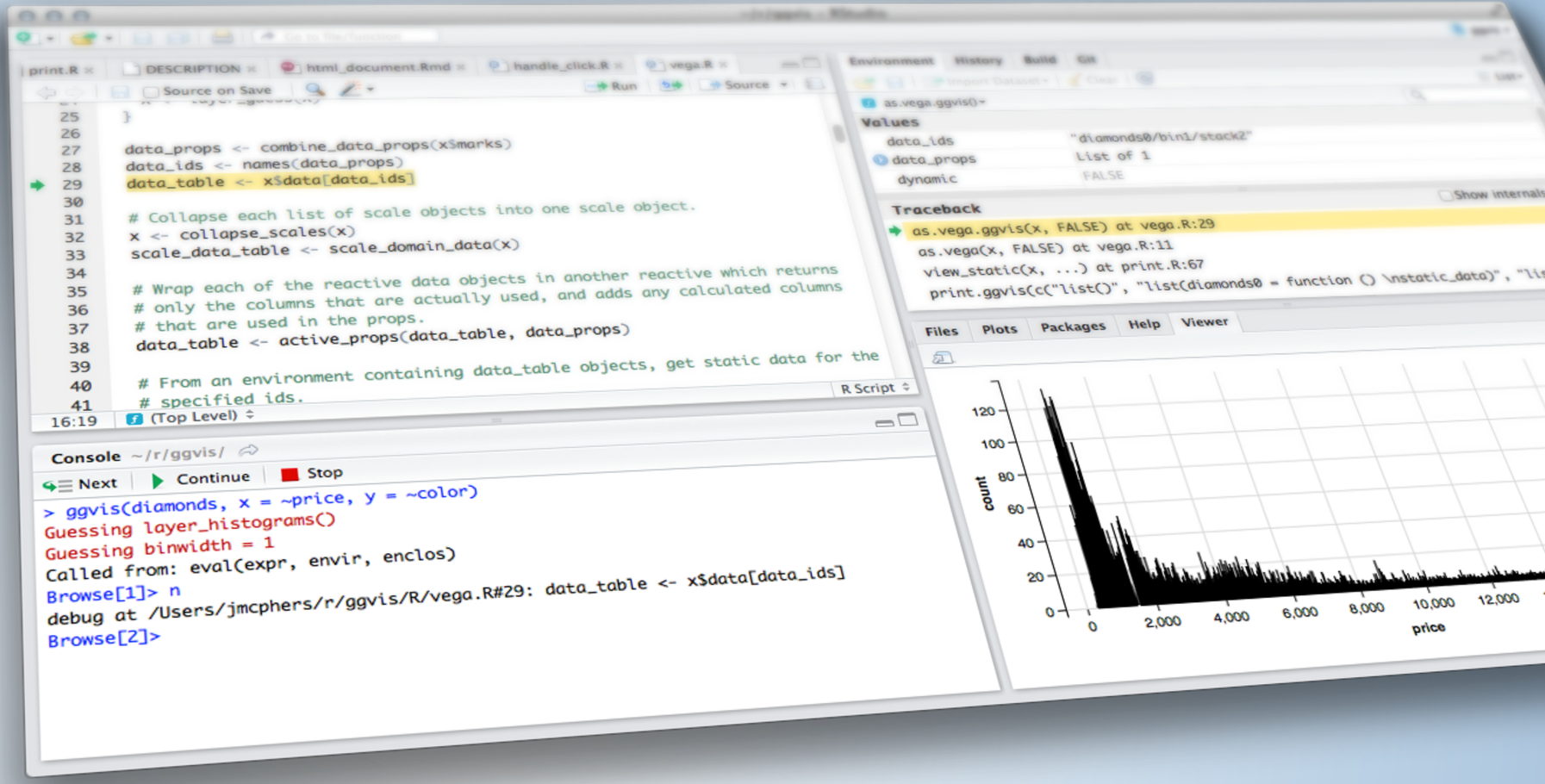


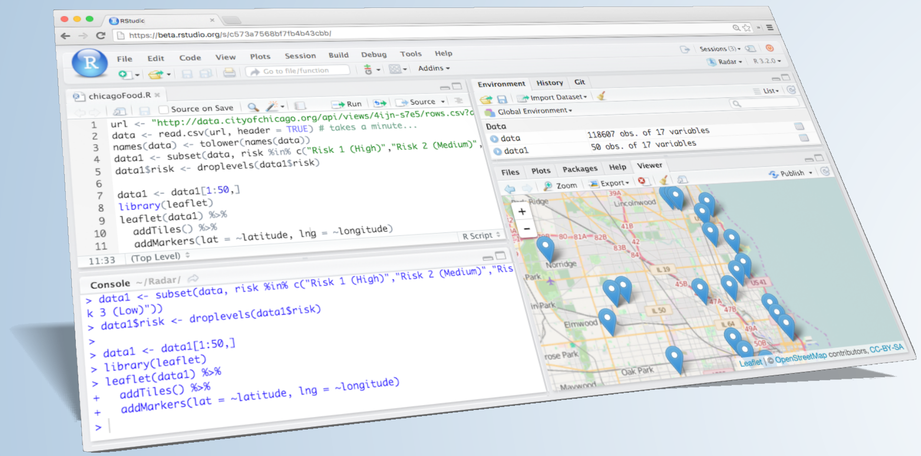
WHAT YOU SHOULD KNOW ABOUT ADMINISTERING RSTUDIO SERVER PRO

SUPPORTING A SCALABLE ENVIRONMENT FOR R

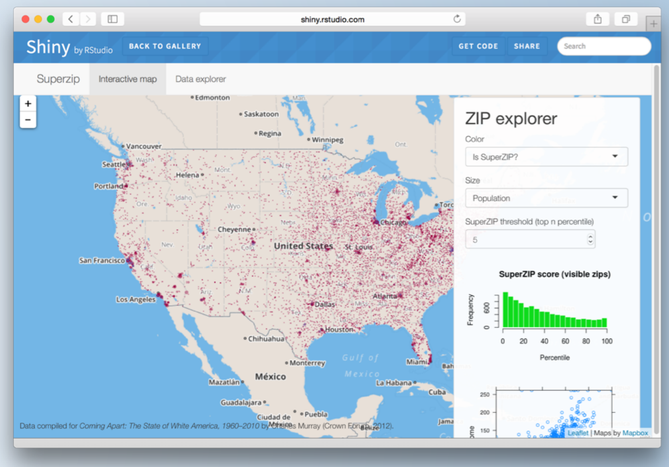
RStudio®

RStudio Products

RStudio IDE



Shiny



Packages

RStudio IDE

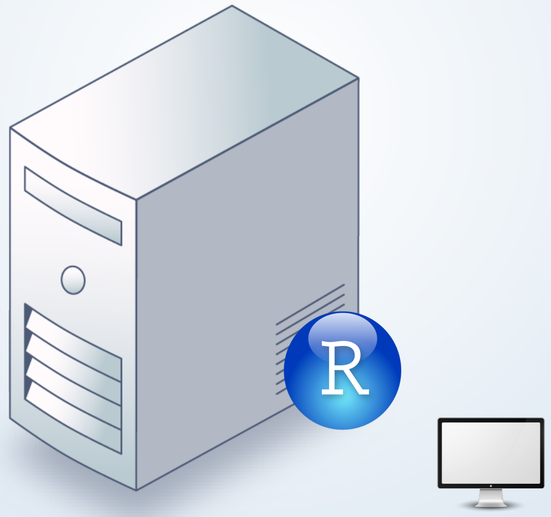
RStudio Desktop

Open Source

Windows / Mac / Linux

RStudio Server

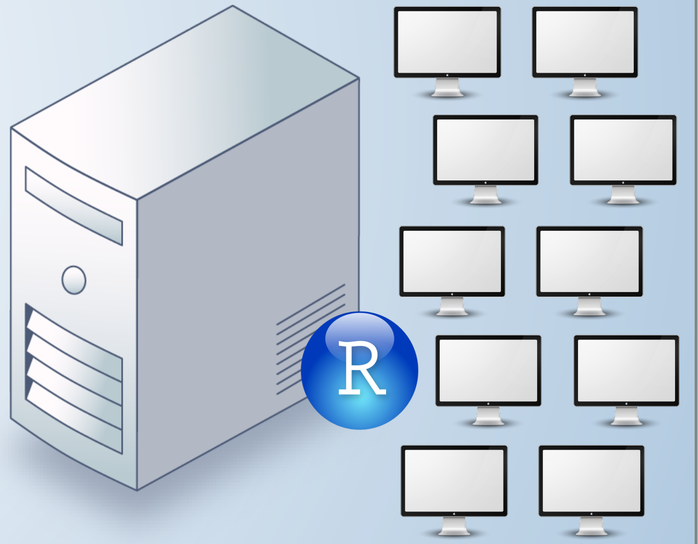
Open Source



Linux

RStudio Server

Professional



Linux

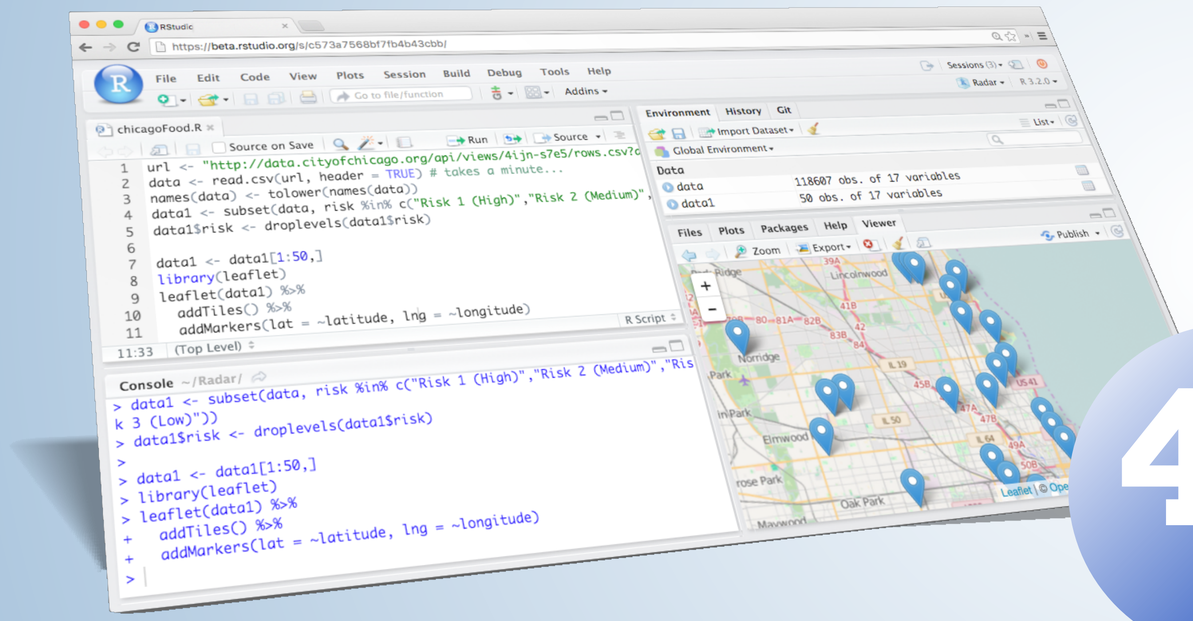
RStudio Server Pro

Make it easier for your teams to use R at scale

Delivers the team productivity, security, centralized management, metrics, and commercial support that professional data science teams need to develop at scale.

RStudio Server Pro Evaluation

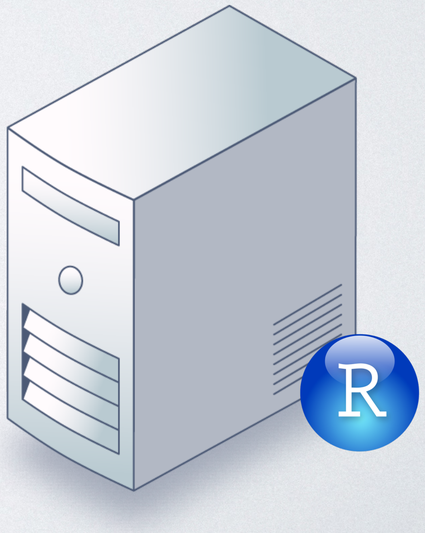
https://www.rstudio.com/products/rstudio-server-pro/evaluation/



45 Day Free Eval

& Access to priority email support

FREQUENTLY ASKED QUESTIONS



What is the difference between R and RStudio

Open Source R

Environment for statistical computing and graphics

R community contributes add-on packages

Packages available for a variety of specific purposes

R Core Team modifies source code

RStudio

Enterprise-ready and open source pro products

Enable teams to **scale, share** work, and be **productive**

Contribute to **open source** R packages and products

Platinum member of the **R Consortium**

What is the difference between your desktop and server products

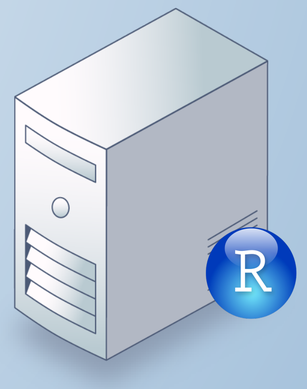
Access from anywhere via the browser

Move computation close to the data

Scale compute and RAM centrally

Process data securely behind the firewall

Administer a central environment



Can I install RStudio Server Pro on a Virtual Machine or in the cloud?

What are the basic requirements?

Operating Systems

Debian/Ubuntu

Redhat/CentOS

openSUSE/SLES

R Engines

CRAN (common)

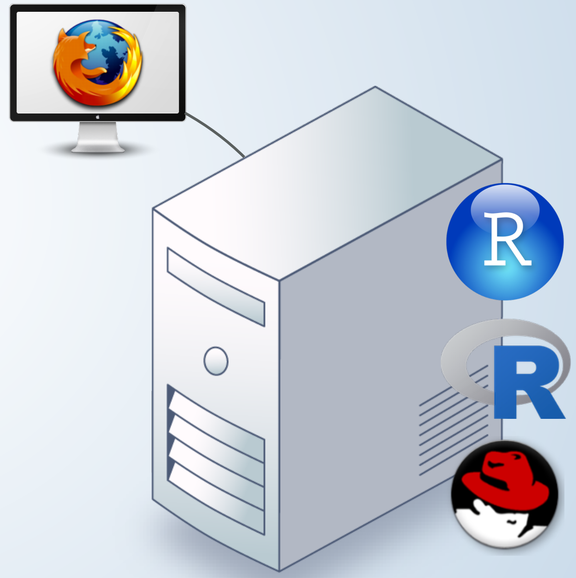
Microsoft, Tibco, Oracle, etc.

Web browsers

Chrome, Firefox, Safari, IE

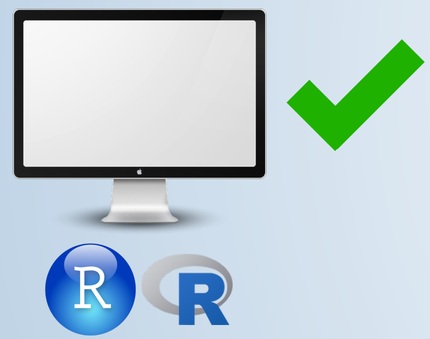
RStudio Server

Example

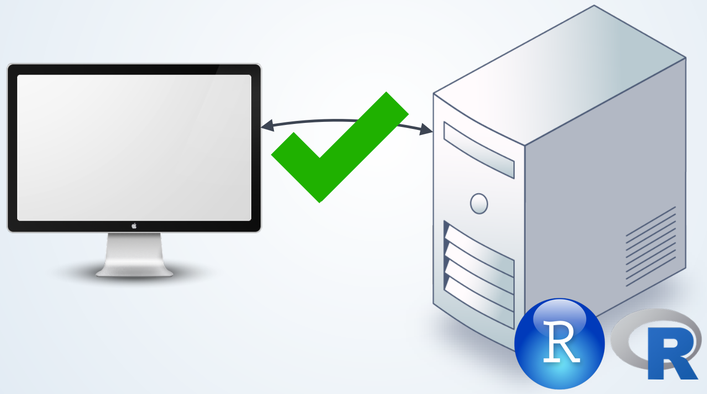


Can you put R and RStudio on different machines?

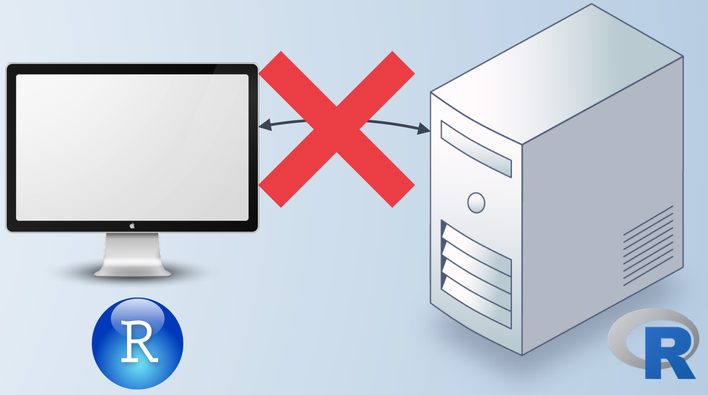
Desktop:
RStudio & R



Server:
RStudio & R

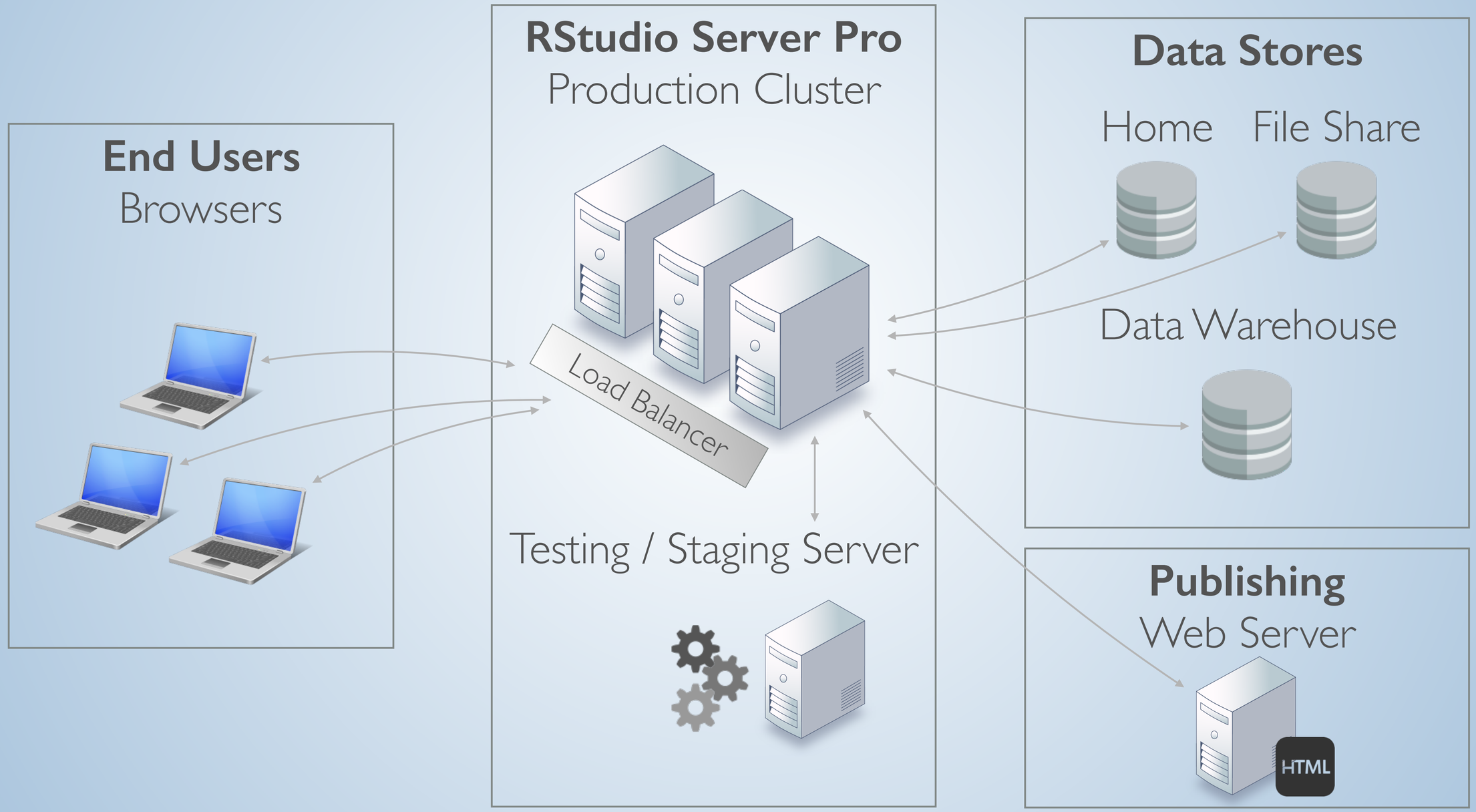


Desktop: RStudio
Server: R

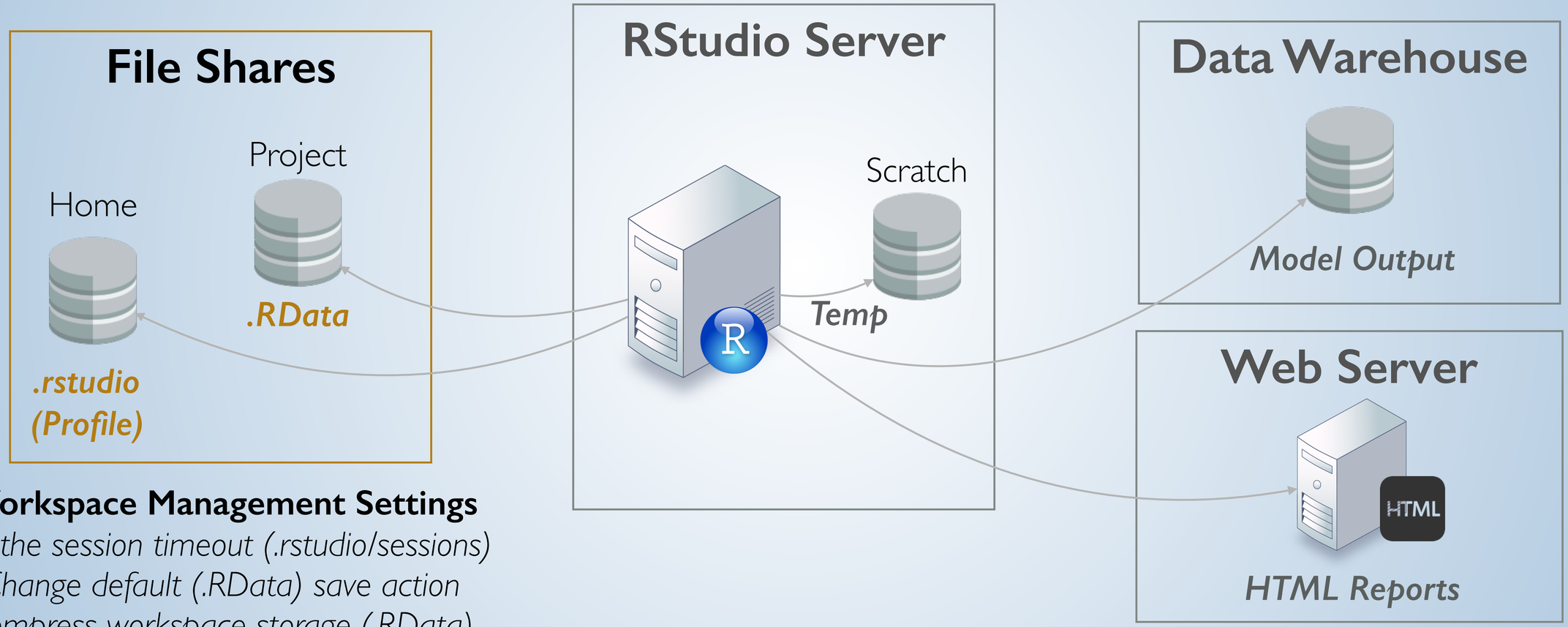


R and RStudio go together.
They must be installed on the same machine.

What does a typical setup look like?



Where does RStudio Store data?



Workspace Management Settings

Set the session timeout (.rstudio/sessions)
Change default (.RData) save action
Compress workspace storage (.RData)

How big should my server be?

https://gallery.shinyapps.io/instanceCalc/

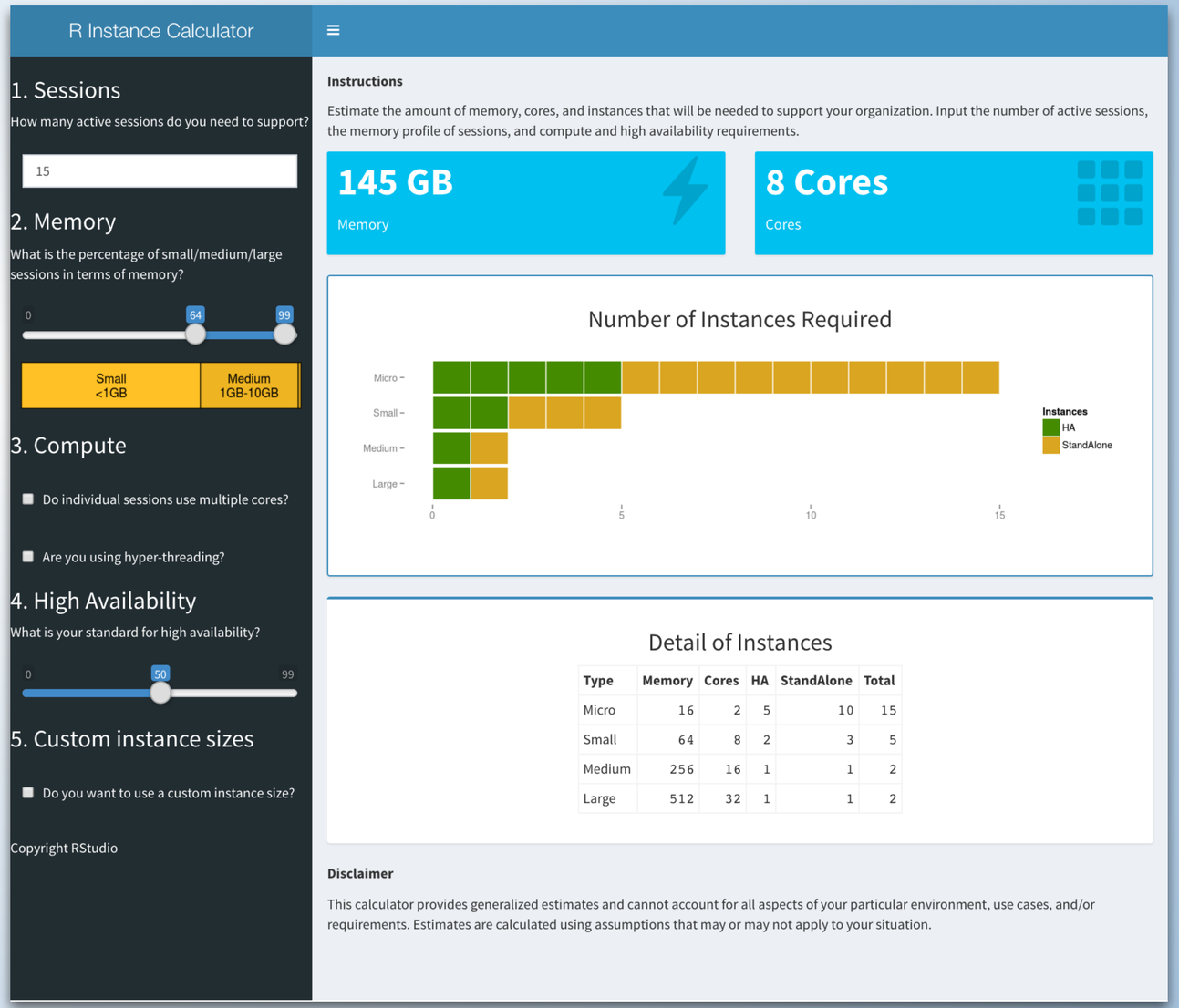
Primary Drivers

1. Number of concurrent sessions
2. Size of sessions

	RAM (GB)
• Small	< 1
• Medium	1 - 10
• Large	10 +

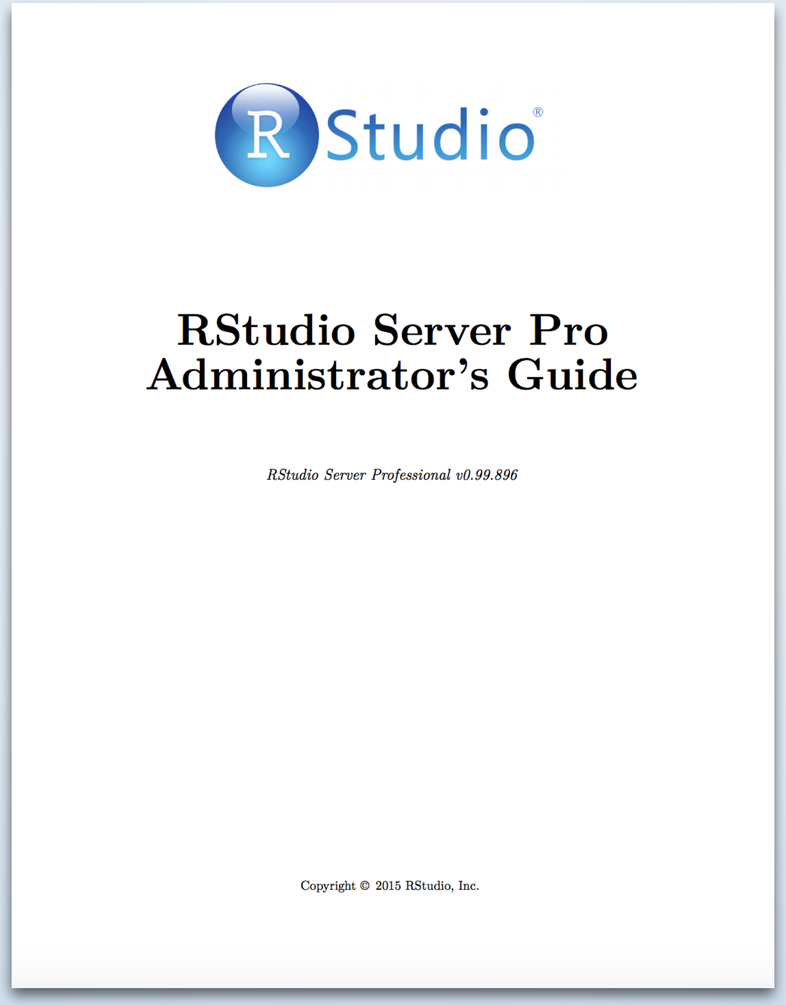
Instance Sizes Examples

	Cores	RAM (GB)
• Micro	2	16
• Small	8	64
• Medium	16	256
• Large	32	512



Where do I go to get information?

http://docs.rstudio.com/



What happens if I run into problems?

Check installation

Check error logs

1.5.2 Troubleshooting Problems

If you are unable to access the server after installation, you should run the `verify-installation` command to output additional diagnostics:

```
$ sudo rstudio-server verify-installation
```

This command will start the server and run and connect to an R session. Note that this will test the correct installation of RStudio Server and ensure that it can connect to a locally installed version of R. However, it won't test whether networking or authentication problems are preventing access to the server.

If problems persist, you can also consult the system log to see if there are additional messages there. On Debian/Ubuntu systems this will typically be located at:

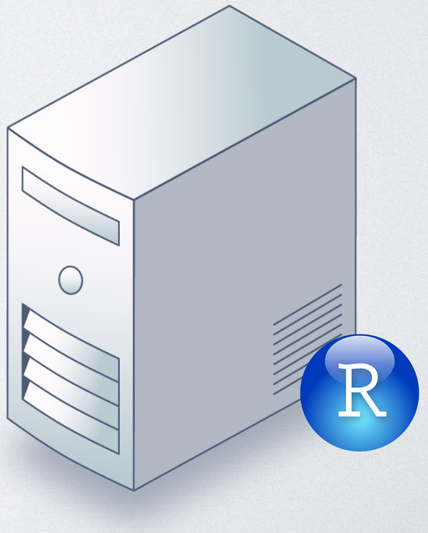
```
/var/log/syslog
```

On RedHat/CentOS systems this will typically be located at:

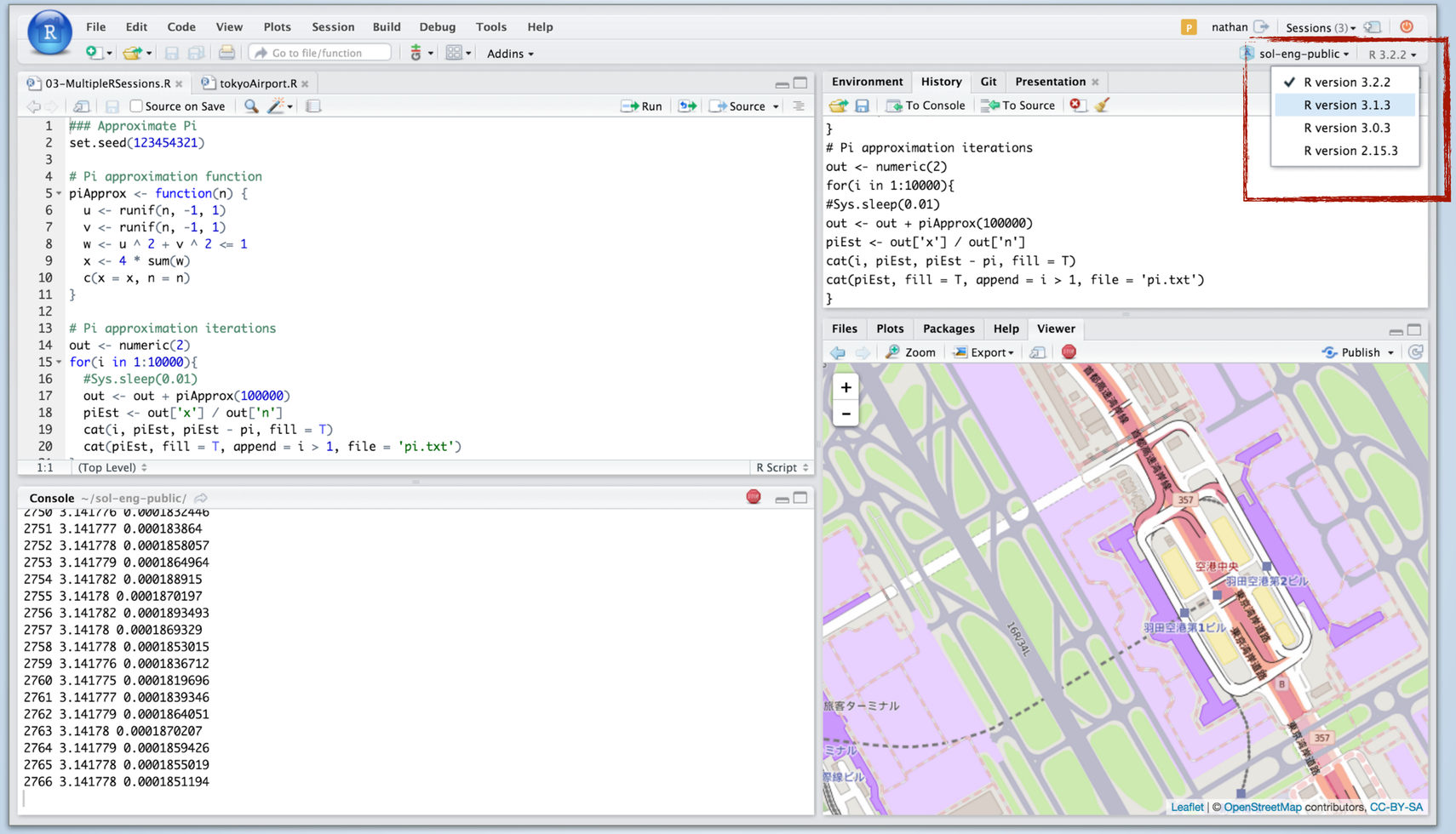
```
/var/log/messages
```

Email support: support@rstudio.com

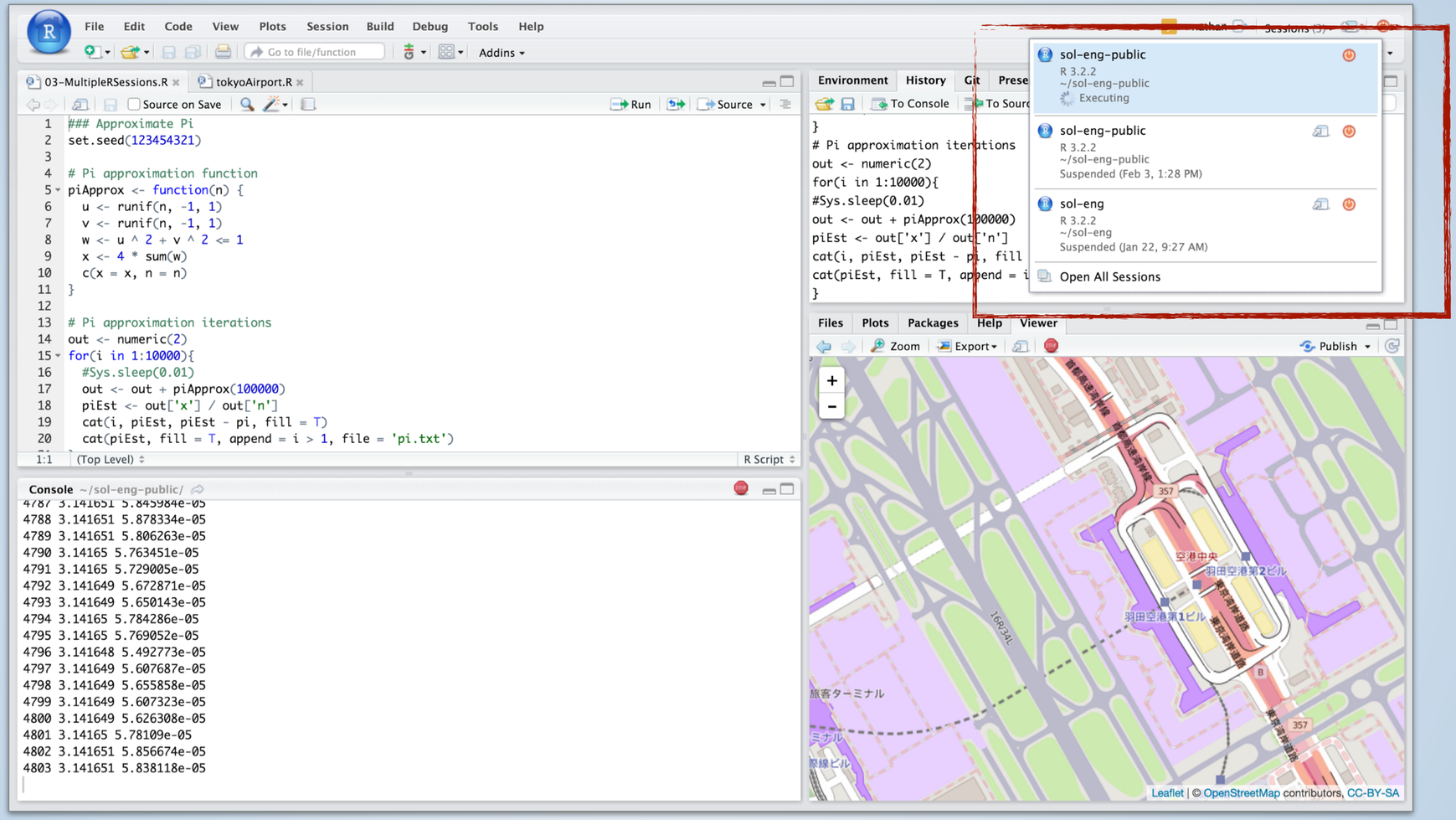
RSP NEW FEATURES



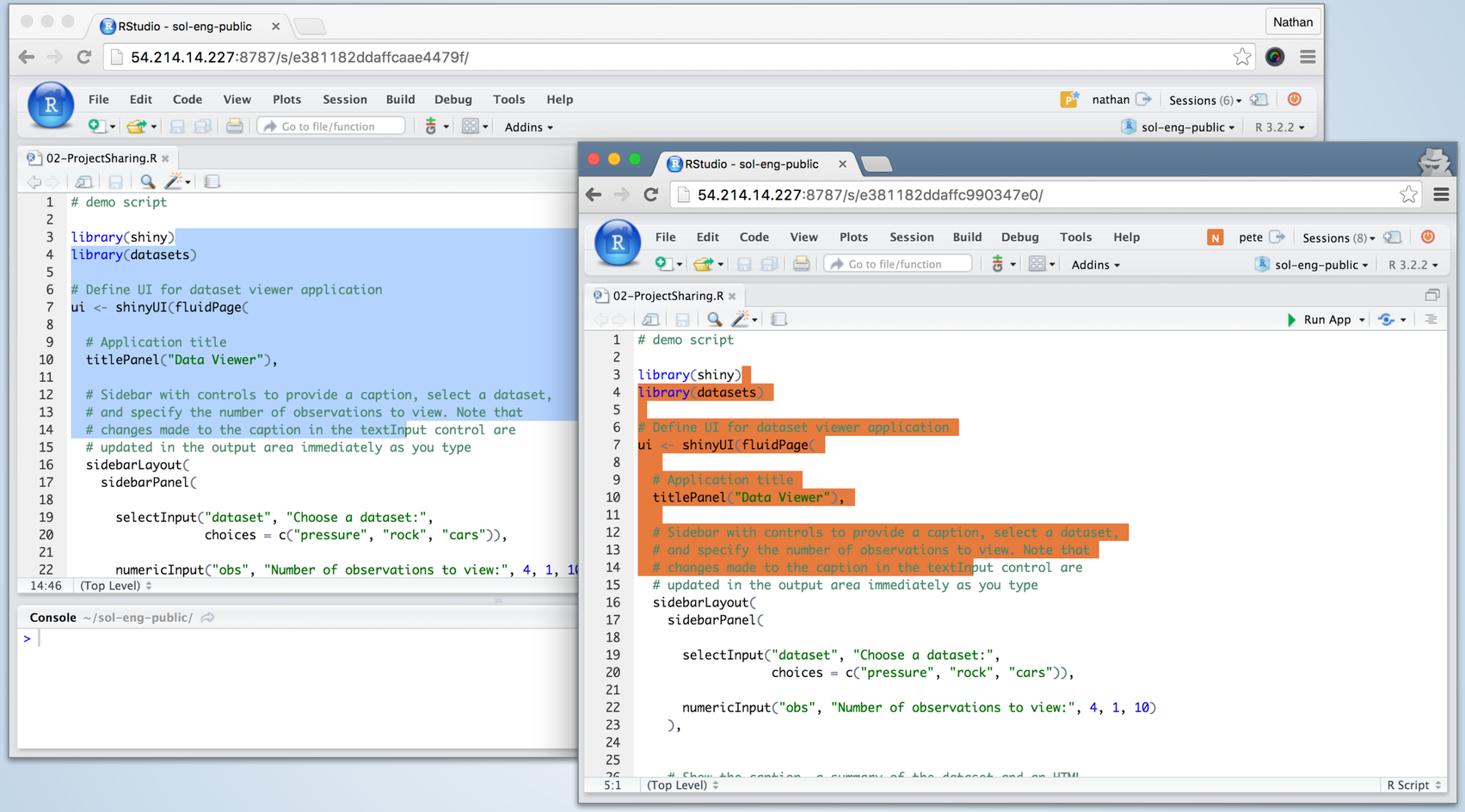
Multiple Versions of R



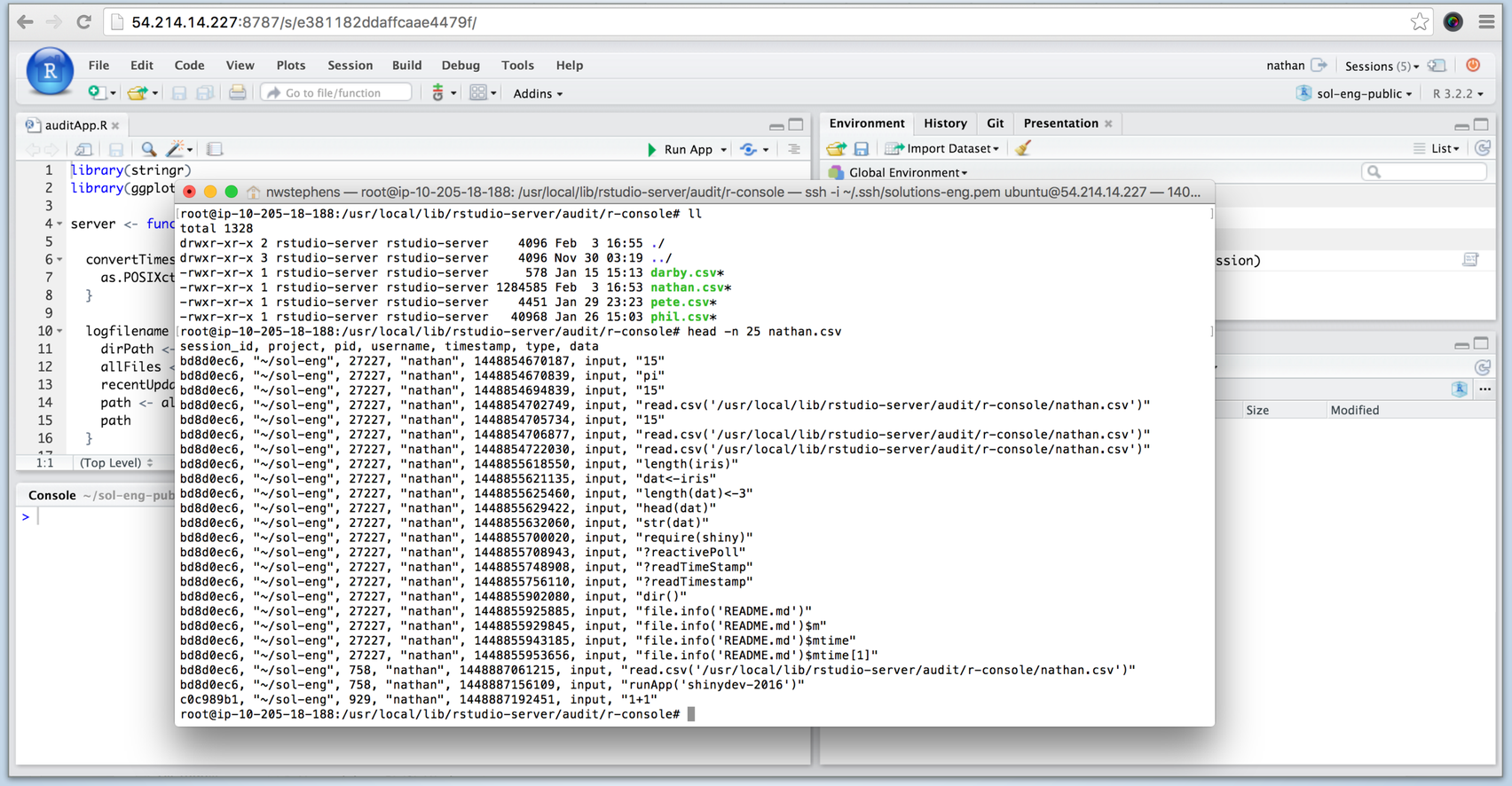
Multiple R Sessions



Project Sharing and Collaborative Editing

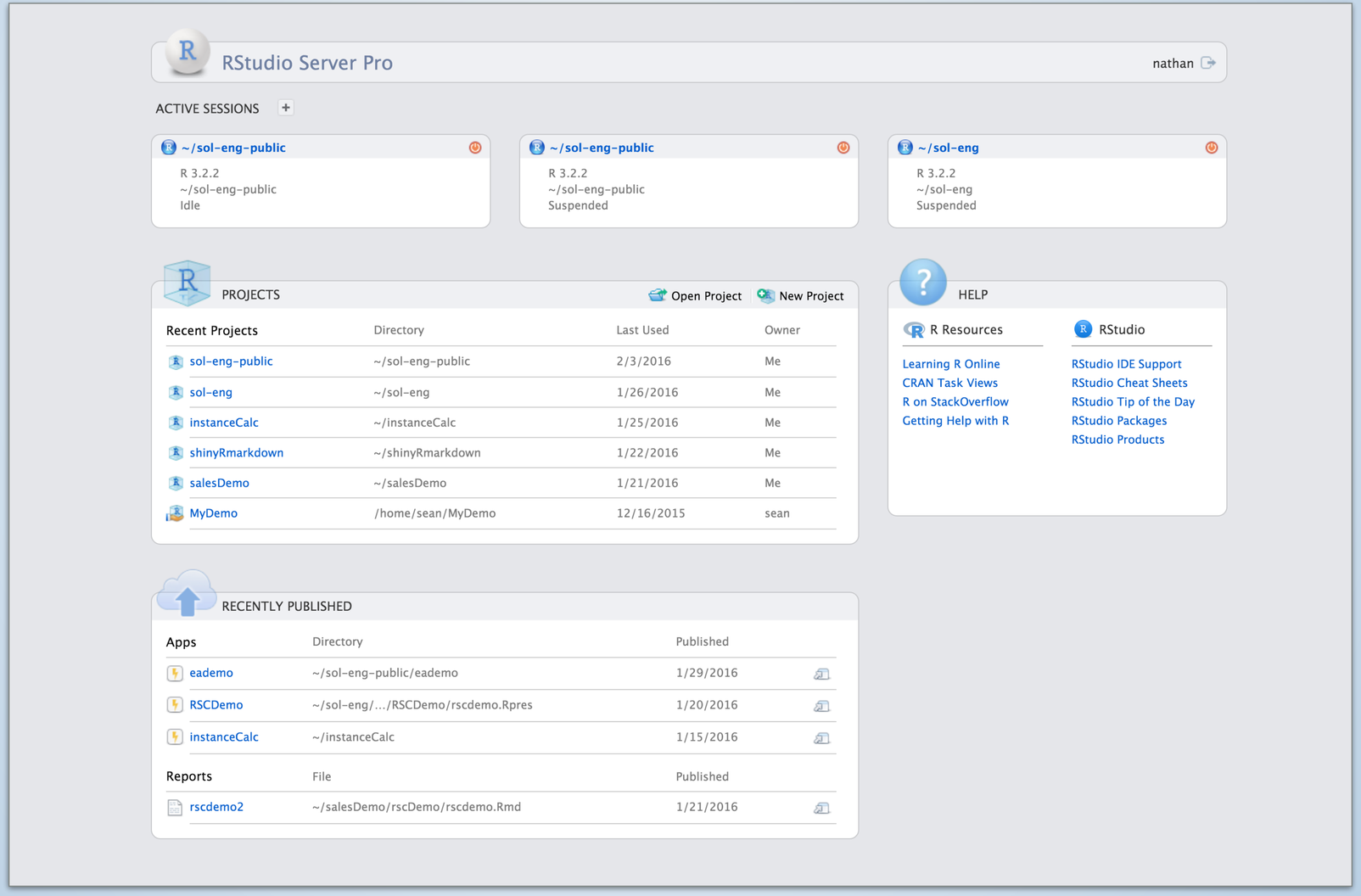


Auditing and Monitoring



RStudio Server Pro Home

Navigate to active sessions, projects, apps and reports



Details: Building R from Source

Identify Linux dependencies

Download source code for R

Configure; Make; Install

Option [Recommended]:
Link to system BLAS libraries

6.2.2 Building Additional Versions from Source

6.2.2.1 Installing Dependencies

Installing additional versions of R side-by-side with the system version requires building R from source but is very straightforward. First, ensure that you have the build dependencies required for R. On RedHat/CentOS you'd use this command:

```
$ sudo yum-builddep R
```

On Debian/Ubuntu systems you'd use this command:

```
$ sudo apt-get build-dep r-base
```

6.2.2.2 Configuring and Building R

Once you've satisfied the build dependencies, you should obtain and unarchive the source tarball for the version of R you want to install. Then from within the extracted source directory execute these commands (this example assumes you are installing R 3.2.0):

```
$ ./configure --prefix=/opt/R/3.2.0 --enable-R-shlib
$ make
$ sudo make install
```

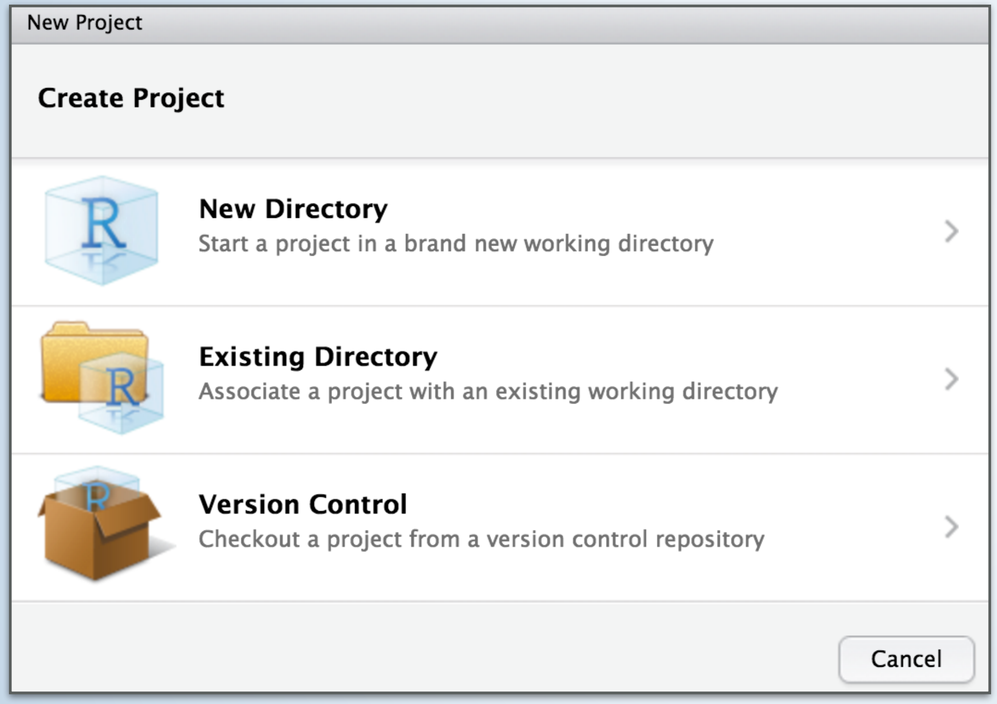
Note that the `--enable-R-shlib` option is required in order to make the underlying R shared library available to RStudio Server.

6.2.2.3 Using the System BLAS Libraries

You may also wish to link to the system BLAS libraries rather than use the R internal versions. For this you'd use the following configure command:

```
./configure --prefix=/opt/R/3.1.0 --enable-R-shlib --with-blas --with-lapack
```

Details: RStudio Projects



Project data
.RData

Project metadata
.RProj

Example Projects
Build packages
Clone repositories
Collaborate and share

Details: Project Sharing

Collaborators are granted same access as owners.

Access Control Lists (ACL's)

Works for load balanced clusters

Can be disabled if desired

5.6.2.1 Access Control Lists

To use Project Sharing, the directories hosting the projects to be shared must be on a volume that supports POSIX Access Control Lists (ACLs). RStudio Server uses ACLs to grant collaborators access to shared projects; ordinary file permissions are not modified.

Instructions for enabling ACLs vary by Linux distribution and filesystem type (see the Guide to enabling ACLs on Ubuntu or RedHat, for example). Broadly, you will need to ensure that the filesystem is mounted with the `user_xattr` and `acl` attributes, and modify `/etc/fstab` if necessary to persist the attributes.

Note that many Linux distributions now have ACLs enabled by default in which case no special configuration is required. You can use the `tune2fs` command to inspect the attributes with which your filesystem is mounted (`user_xattr` and `acl` are required for project sharing).

5.6.2.2 Project Sharing and NFS

If you plan to use Project Sharing with NFS-mounted volumes, there are several caveats you should be aware of.

1. We recommend using NFSv3 instead of NFSv4, since NFSv4 has its own ACL model which (despite being richer) doesn't inter-operate well with POSIX ACLs. For more information, see ACL Interoperability with NFS. You can typically use the `nfsvers=3` mount option to ensure your client connects with version 3 of the protocol.
2. We recommend mounting NFS with the `noac` mount option. Without this option, NFS caches file attributes, so it may not be possible for users working simultaneously in a project to know whether they're seeing each others' latest changes. The `noac` option does reduce performance, however, so we recommend testing to choose the right trade off for your environment.
3. NFS must be mounted with the `acl` mount option. On most systems this is the default, so you need only ensure that `noacl` is not present. It's also necessary for the file system on the NFS server to itself be appropriately configured for ACL support; see the section above on Access Control Lists for guidance.
4. Some features which automatically update when directory contents change will not update on NFS. For instance, users may need to manually refresh the Files pane to see new files added by collaborators.

Details: Auditing

Capture input and output

Customize file location and size

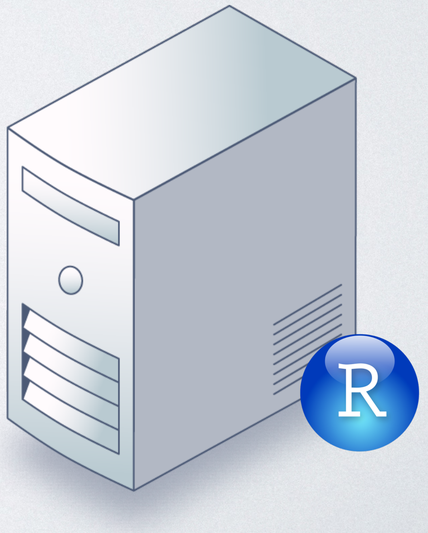
8.1.2 Data Format

The R console activity for each user is written into individual files within the `r-console` data directory (by default `/var/lib/rstudio-server/audit/r-console`). The following fields are included:

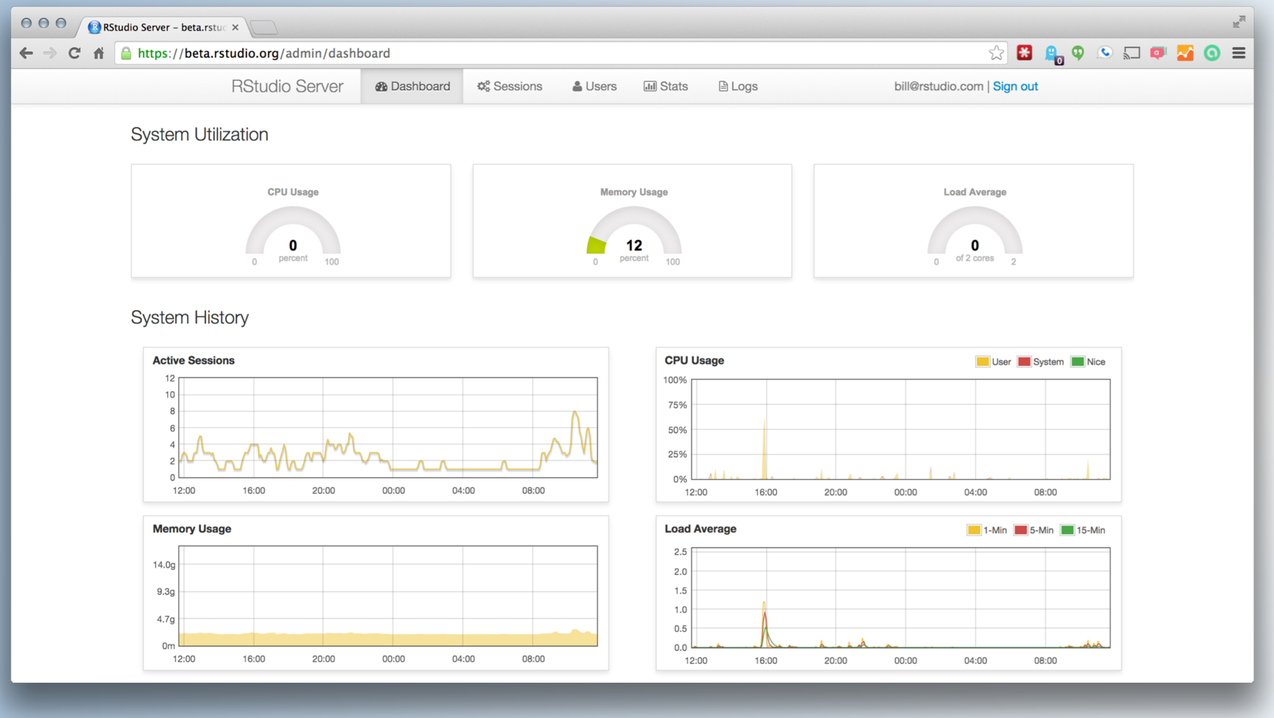
session_id	Unique identifier for R session where this action occurred.
project	Path to RStudio project directory if the action occurred within a project.
pid	Unix process ID where this console action occurred.
username	Unix user which executed this console action.
timestamp	Timestamp of action in milliseconds since the epoch.
type	Console action type (prompt, input, output, or error).
data	Console data associated with this action (e.g. output text).

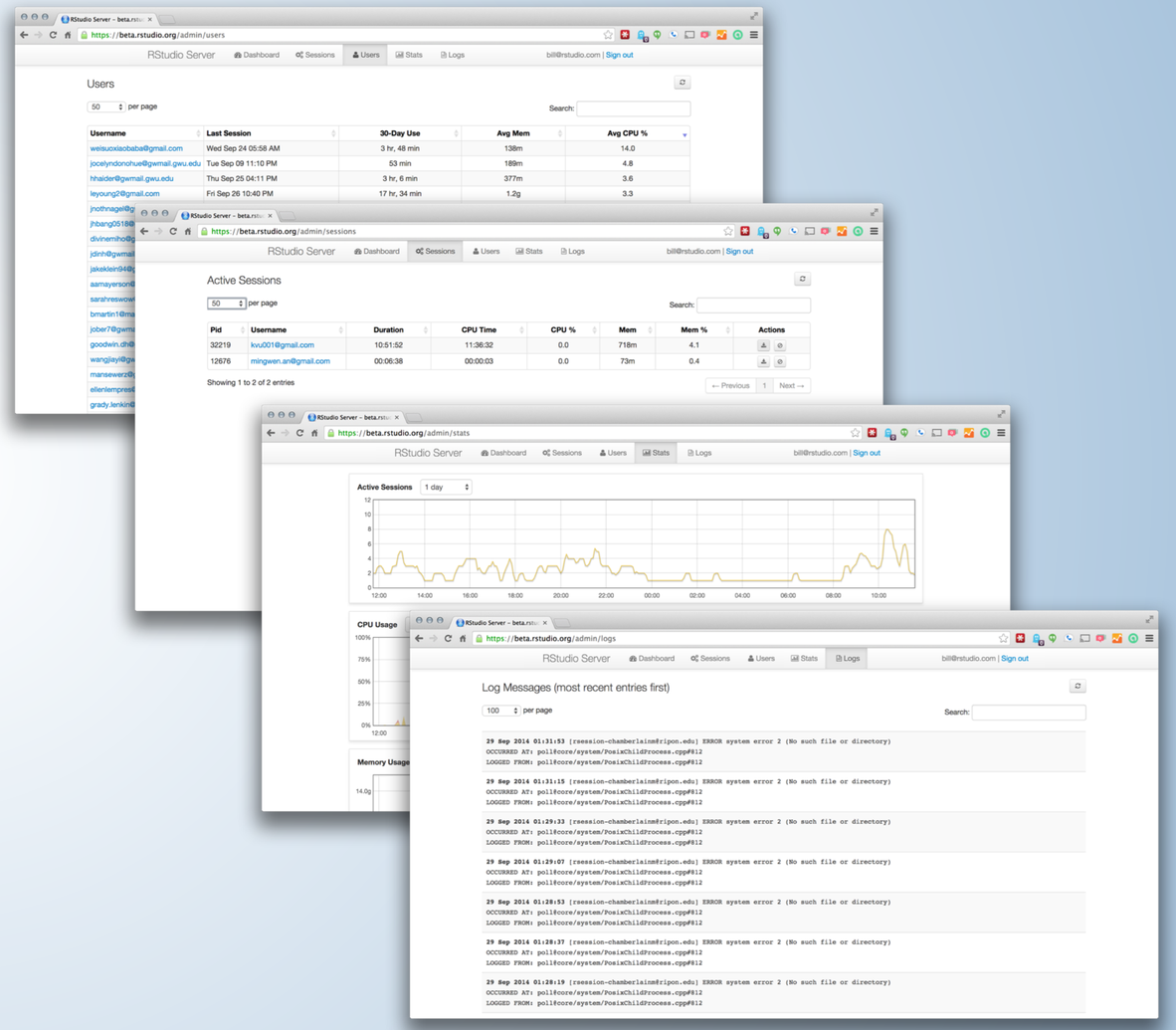
The `session_id` field refers to a concurrent R session as described in the section on Multiple R Sessions (i.e. it can span multiple projects and/or pids).

CONTROL YOUR ENVIRONMENT



Administrative Dashboards





Admin-group: Can view dashboards
Admin-superuser-group: Can suspend, terminate, and assume control of sessions

Monitoring and Health Checks

Monitoring is written to a set of RRD files and can be sent to Graphite or Carbon compatible system.

```
monitor-graphite-enabled=1
monitor-graphite-host=134.47.22.6
```

If you are using a service like hostedgraphite.com that requires that you provide an API key as part of reporting metrics you can use the **monitor-graphite-client-id** setting. For example:

/etc/rstudio/rserver.conf

```
monitor-graphite-enabled=1
monitor-graphite-host=carbon.hostedgraphite.com
monitor-graphite-client-id=490662a4-1d8c-11e5-b06d-000c298f3d04
```

You can poll RStudio Server for health checks

```
active-sessions: 1
cpu-percent: 0.0
memory-percent: 64.2
swap-percent: 0.0
load-average: 4.1
```

Server Management

Start and stop server
Manage active sessions
Kill and suspend sessions
Take the server offline

Use configuration files to apply settings

Upgrades are easy!

Apply license keys to activate

2.1 Core Administrative Tasks

2.1.1 Configuration Files

RStudio Server uses several configuration files all located within the `/etc/rstudio` directory. Configuration files include:

rserver.conf	Core server settings
rsession.conf	Settings related to individual R sessions
profiles	User and group resource limits
r-versions	Manual specification of additional versions of R
ip-rules	IP access rules (allow or deny groups of IP addresses)
load-balancer	Load balancing configuration
health-check	Template for content to return for server health checks
google-accounts	Mappings from Google accounts to local accounts

The `rserver.conf` and `rsession.conf` files are created by default during installation however the other config files are optional so need to be created explicitly.

```
$ sudo gdebi <rstudio-server-package.deb>
```

For RedHat/CentOS:

```
$ sudo yum install --nogpgcheck <rstudio-server-package.rpm>
```

For openSUSE / SLES:

```
$ sudo zypper install <rstudio-server-package.rpm>
```

```
$ sudo rstudio-server license-manager status
```

To activate the product you obtain a product key and then use the following commands:

```
$ sudo rstudio-server license-manager activate <product-key>
$ sudo rstudio-server restart
```

User and Group Profiles

Tailor the behavior of R on a per-user or per-group basis

1. Version of R
2. CPU affinity
3. Scheduling priority (i.e. nice value)
4. Resource limits
    1. Maximum memory
    2. Max processes
    3. Max open files

Automatically execute user profile scripts

5.2.1 Creating Profiles

Profiles are defined within the file `/etc/rstudio/profiles`. Note that this file is not created by default so you'll need to create it if doesn't already exist. Profiles are divided into sections of three different type:

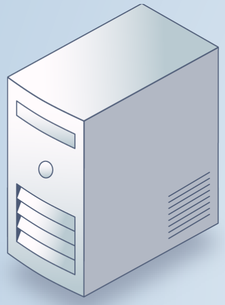
1) Global (`[*]`)
2) Per-group (`[@groupname]`)
3) Per-user (`[username]`)

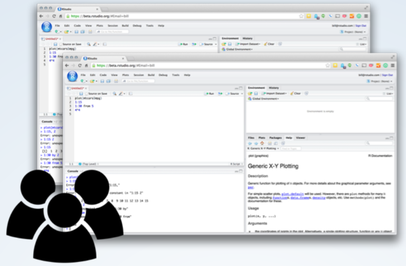
Here's an example profiles file that illustrates each of these types:

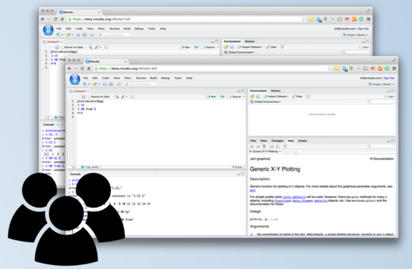
/etc/rstudio/profiles

```
[*]
cpu-affinity = 1-4
max-processes = 100
max-memory-mb = 2048

[@powerusers]
cpu-affinity = 5-16
nice = -10
max-memory-mb = 4096

[jsmith]
r-version = /opt/R/3.1.0
```

User and Group Profiles Example







	Compliance Team	Power User Group
R version	3.0.3	3.2.3
CPU cores	1 - 4	5 - 8
Scheduling Priority	10	-10
Max Processes	50	200
Max Memory	4 GB	16 GB

Feature Limits

Limit file size uploads

Limit CPU time per computation

Notify when disk quota is exceeded

5.8.2 Maximum File Upload Size

You can limit the maximum size of a file upload by using the `limit-file-upload-size-mb` setting. For example, the following limits file uploads to 100MB:

/etc/rstudio/rsession.conf

```
limit-file-upload-size-mb=100
```

The default behavior is no limit on the size of file uploads.

5.8.3 CPU Time per Computation

If you want to prevent runaway computations that consume 100% of the CPU you can set the maximum number of minutes to allow top-level R computations to run for using the `limit-cpu-time-minutes` setting. For example:

/etc/rstudio/rsession.conf

```
limit-cpu-time-minutes=30
```

This specifies that no top level computation entered at the R console should run for more than 30 minutes. This constraint is implemented by calling the R `setTimeLimit` function immediately prior to handing off console commands to R. As a result it is possible for a particular script to override this behavior if it knows that it may exceed the threshold. This would be done as follows:

```
setTimeLimit(cpu = Inf)
# Long running R code here...
```

5.8.4 XFS Disk Quotas

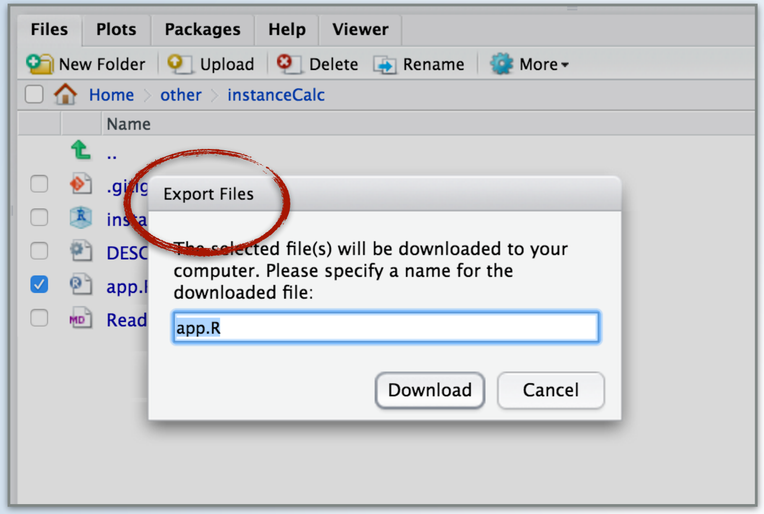
If your system uses the XFS file system (http://en.wikipedia.org/wiki/XFS) then RStudio Server can be configured to notify users when they come close to or exceed their disk quota. You can enable this using the `limit-xfs-disk-quota` setting. For example:

/etc/rstudio/rsession.conf

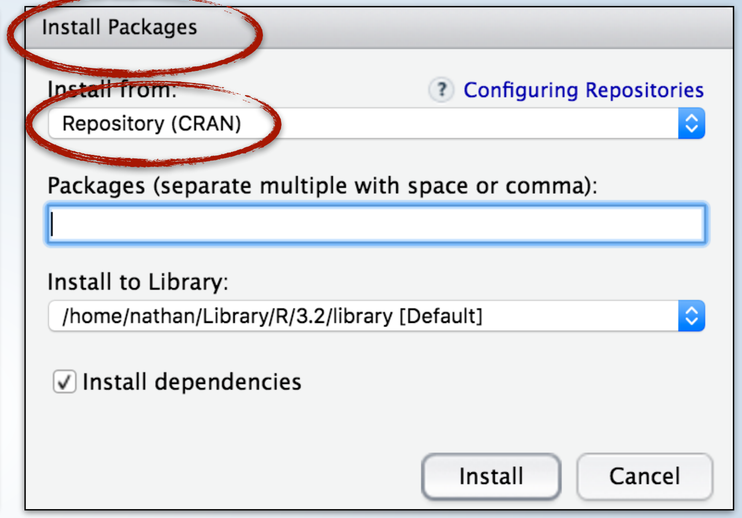
```
limit-xfs-disk-quota=1
```

Turning Off Feature Limits

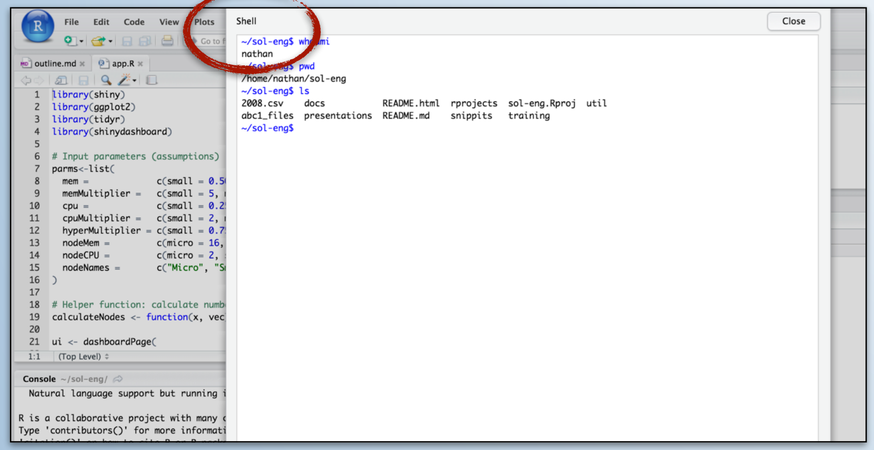
Export Files



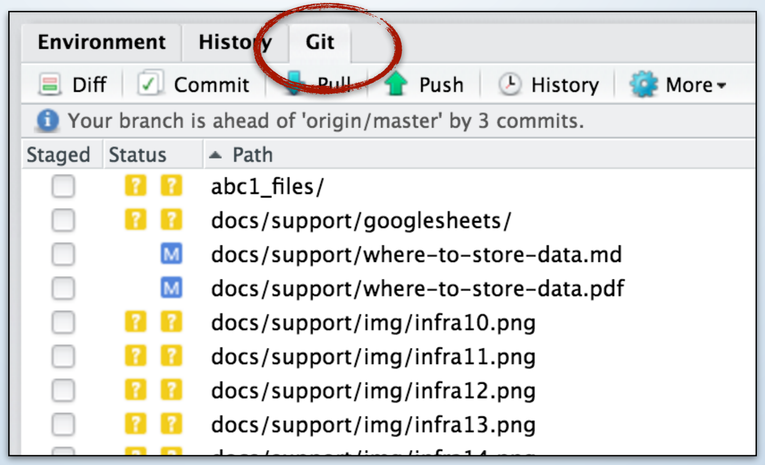
Install Packages



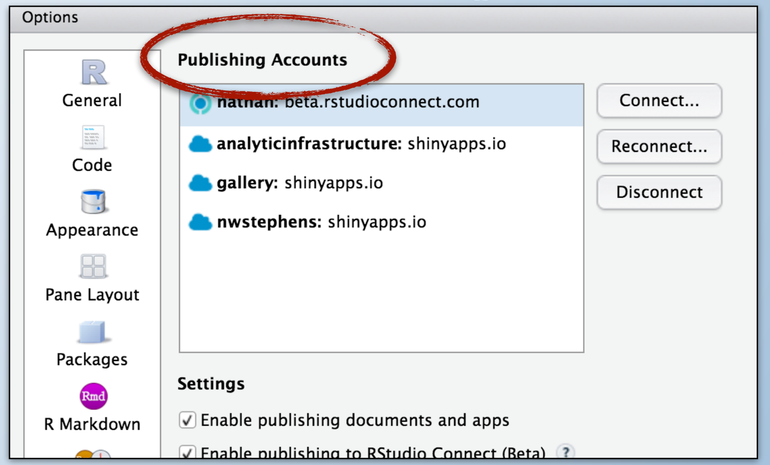
Shell



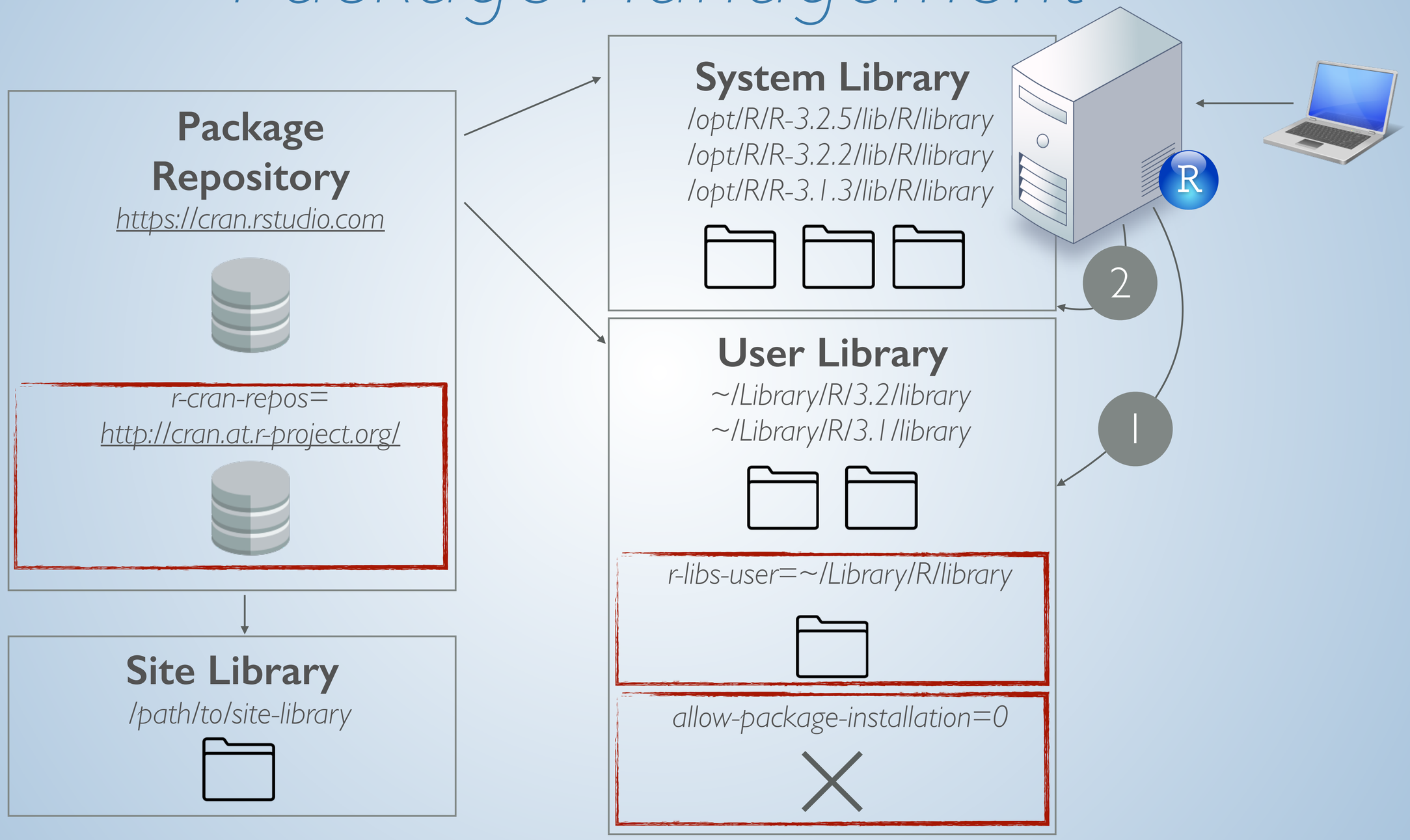
Git



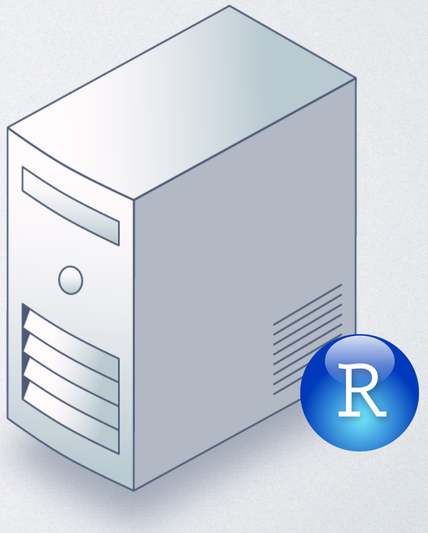
Publishing



Package Management



PROTECT YOUR ENVIRONMENT



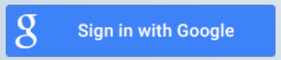
Authenticating Users

System Accounts

PAM

- Active Directory
- LDAP
- Kerberos tickets

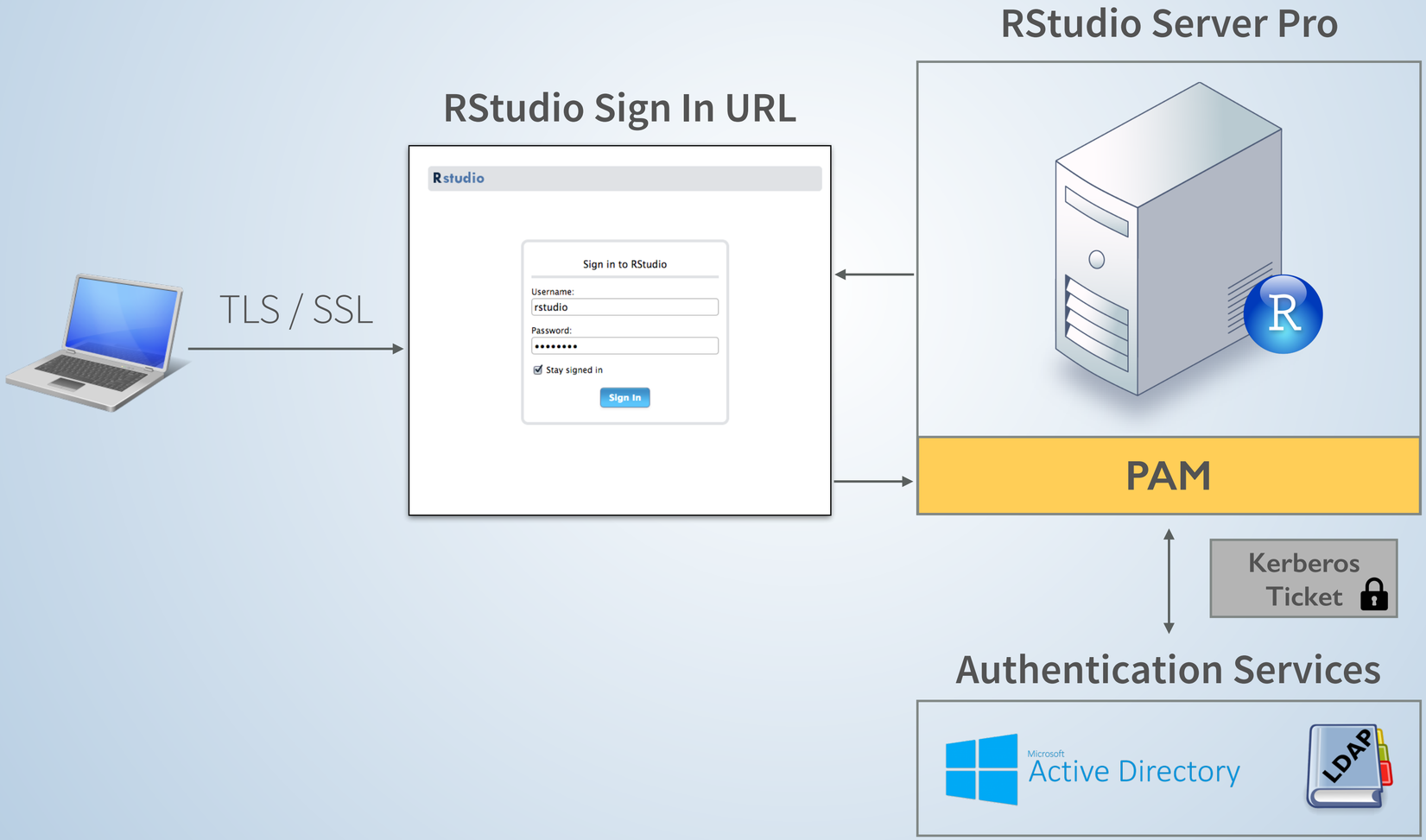
Proxied



Google

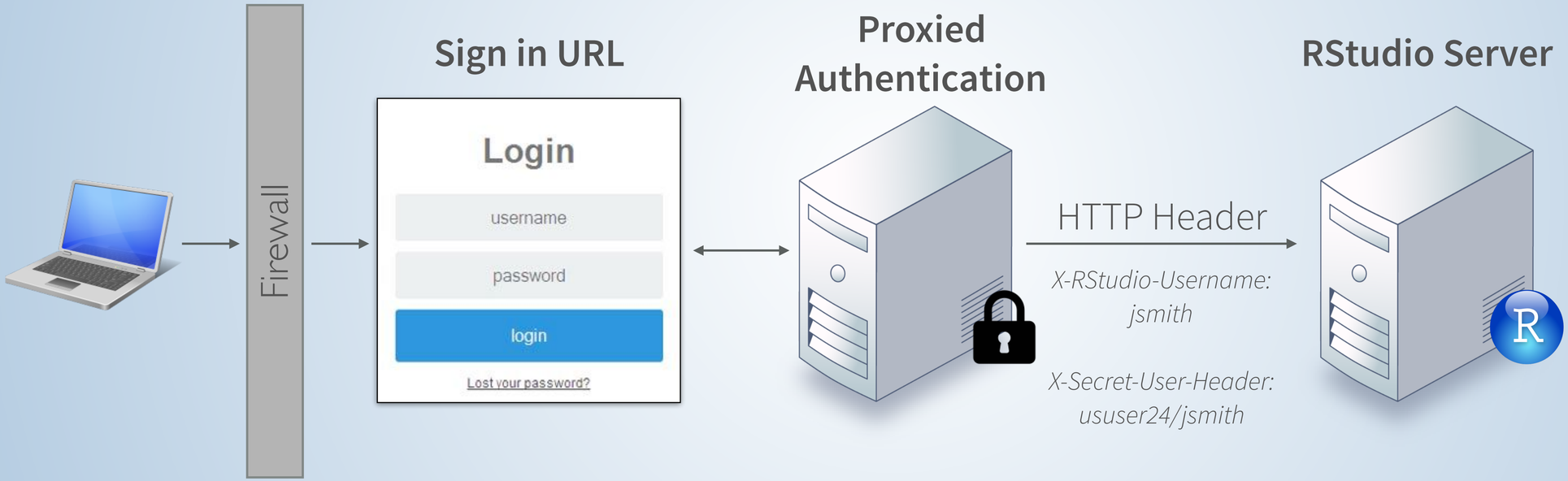
Authenticating Users

PAM Authentication



Authenticating Users

Proxied Authentication / Single Sign On



Enable proxy authentication

3.4.1 Enabling Proxied Authentication

To enable proxied authentication you need to specify both the `auth-proxy` and `auth-proxy-sign-in-url` settings (the sign-in URL is the absolute URL to the page that users should be redirected to for sign-in). For example:

/etc/rstudio/rserver.conf

```
auth-proxy=1
auth-proxy-sign-in-url=http://example.com/sign-in
```

Access and Security

Modify the port (default is 8787)

Deny / allow specific IP's or IP ranges

Enable SSL (TLS V1, V1.1, V1.2)

4.1 Network Port and Address

After initial installation RStudio accepts connections on port 8787. If you wish to listen on a different another port you can modify the **www-port** option. For example:

/etc/rstudio/rserver.conf

```
www-port=80
```

4.2 IP Access Rules

RStudio Server can be configured to deny access to specific IP addresses or ranges of addresses. Access rules are defined in the configuration file **/etc/rstudio/ip-rules**

Access rules are established using the **allow** and **deny** directives and are processed in order, with the first matching rule governing whether a given address is allowed or denied. For example, to allow only clients within the 192.168.1.0/24 subnet but also deny access to 192.168.1.10 you would use these rules:

/etc/rstudio/ip-rules

```
deny    192.168.1.10
allow   192.168.1.0/24
deny    all
```

4.3.1 SSL Configuration

If your RStudio Server is running on a public network then configuring it to use SSL (Secure Sockets Layer) encryption is strongly recommended. You can do this via the **ssl-enabled** setting along with related settings that specify the location of your SSL certificate and key. For example:

/etc/rstudio/rserver.conf

```
ssl-enabled=1
ssl-certificate=/var/certs/your_domain_name.crt
ssl-certificate-key=/var/certs/your_domain_name.key
```

Server Permissions

Installation of RStudio requires root

RStudio Server runs as root *briefly*
1. During startup
2. When creating R sessions on behalf of users

Server account name
1. Default is *rstudio-server*
2. Can be configured for an alternative account

1.2 Installation

1.2.1 RedHat / CentOS (5+)

Installing R

You can install R for RedHat and CentOS using the instructions on CRAN: https://cran.rstudio.com/bin/linux/redhat/README.

Installation Commands

After downloading the appropriate RedHat/CentOS package for RStudio Server Professional you should execute the following command to complete the installation:

```
sudo yum install --nogpgcheck <rstudio-server-package.rpm>
```

4.4 Server Permissions

4.4.1 Server Account

RStudio Server runs as the system root user during startup and then drops this privilege and runs as a more restricted user. RStudio Server then re-assumes root privilege for a brief instant when creating R sessions on behalf of users (the server needs to call `setresuid` when creating the R session, and this call requires root privilege).

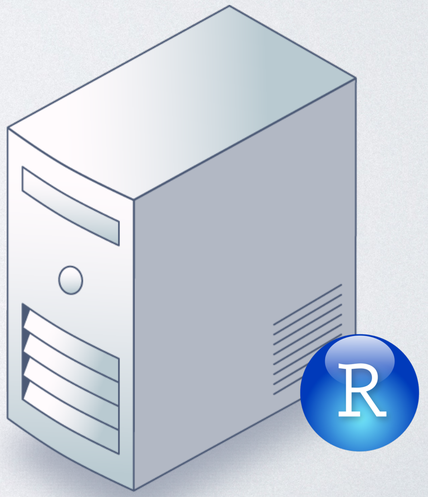
The user account that RStudio Server runs under in the normal course of operations is `rstudio-server`. This account is automatically added to the system during installation and is created as a system rather than end user account (i.e. the `--system` flag is passed to `useradd`).

4.4.1.1 Alternate Server Account

You can configure RStudio Server so that it will run from an alternate account with the following steps:

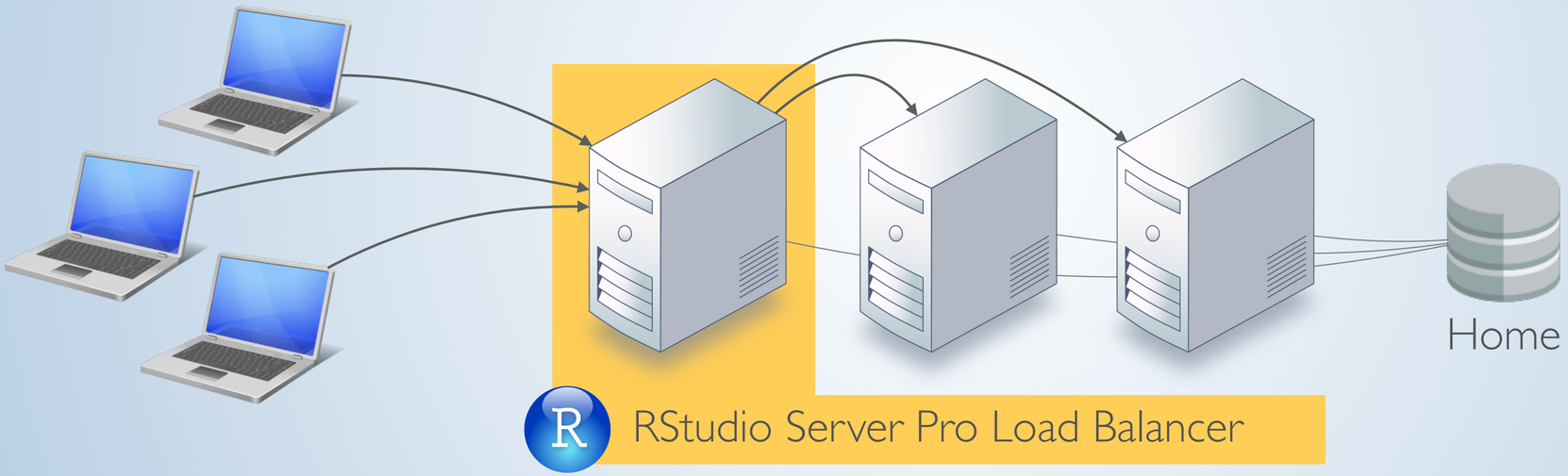
1. Recursively delete the `/var/log/rstudio-server` and `/var/lib/rstudio-server` directories (they contain files and directories owned by the default rstudio server user)
2. Create a new system user (if the one you want to use doesn't already exist)
3. Assign this user to the `server-user` option in the `/etc/rstudio/rserver.conf` configuration file (see example below)
4. Restart RStudio Server

SCALE YOUR ENVIRONMENT



Load Balancing User Sessions

Single Master

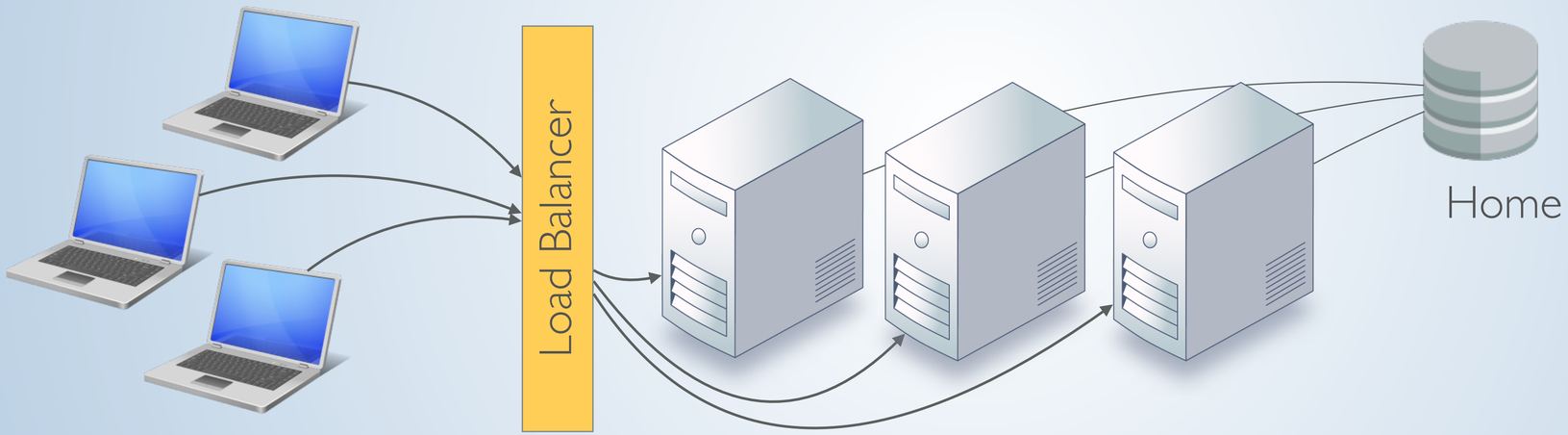


Balancing Methods

SSL Enabled

Load Balancing User Sessions

Multiple Masters



Balancing Methods

SSL Enabled

Load Balancing Examples

Single master configuration

Multiple masters with
Nginx reverse-proxy configuration

7.2.2 Defining Nodes

To define a cluster node, two configuration files need to be provided:

```
/etc/rstudio/load-balancer
/etc/rstudio/secure-cookie-key
```

The first of these defines the available nodes and load balancing strategy. The second defines a shared key used for signing cookies (in single node configurations this key is generated automatically, however with multiple nodes explicit coordination is required).

For example, to define a cluster with two nodes that load balances based the number of actively running R sessions you could use the following configuration:

/etc/rstudio/load-balancer

```
[config]

balancer = sessions

[nodes]

server1.example.com
server2.example.com
```

/etc/rstudio/secure-cookie-key

```
a55e5dc0-d6ae-11e3-9334-000c29635f71
```

For example, here's an Nginx reverse-proxy configuration that you could use with the cluster defined above:

```
http {
  upstream rstudio-server {
    server rstudio1.example.com;
    server rstudio2.example.com backup;
    server rstudio3.example.com backup;
  }
  server {
    listen 80;
    location / {
      proxy_pass http://rstudio-server;
      proxy_redirect http://rstudio-server/ $scheme://$host/;
    }
  }
}
```

RStudio Server Pro Features

New!

- **Project Sharing** ✓
- **Multiple Versions of R** ✓
- **Multiple R Sessions** ✓
- **Auditing** and Monitoring ✓
- Enhanced Security and Authentication ✓
- Load Balancing ✓
- Administrative Dashboard ✓
- Advanced R Session Management ✓

RSTUDIO SERVER PRO

MAKE IT EASIER FOR YOUR TEAMS TO USE R AT SCALE